VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Implementace aplikačního rámce pro deskové hry

Implementation of Framework for Deskboard Games

2012 Lumír Kojecký

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Lumír Kojecký**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Implementace aplikačního rámce pro deskové hry
Implementation of Framework for Deskboard Games

Zásady pro vypracování:

Cílem práce bude zanalyzovat společné rysy a principy deskových her a navrhnout prostředí pro návrh těchto her.

Během řešení postupujte podle následujících bodů:
1. Seznamte se, kategorizujte a popište principy deskových her.
2. Vytvořte analýzu aplikačního rámce.
2. Navrhněte a implementujte tento aplikační rámce, který bude umožňovat definovat parametricky různé deskové hry
3. Otestujte tento rámec v rámci implementace vzorové hry.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Lumír Návrat**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 4. května 2012 .....

V souvislosti s vypracováním bakalářské práce bych rád poděkoval za odbornou a profesionální podporu Ing. Lumíru Návratovi, vedoucímu bakalářské práce.

Abstrakt

Cílem této práce je zanalyzovat společné rysy a principy deskových her a navrhnout prostředí pro návrh těchto her. Práce popisuje deskové hry jako takové, stručně uvádí jejich historii a rozdělení. Práce také popisuje vývojová prostředí/aplikační rámce, které již vytvořeny jsou, a jejich nejdůležitější vlastnosti. Tyto vlastnosti poslouží jako základ pro tvorbu nového aplikačního rámce pro deskové hry. Dále je v práci popsán vývoj tohoto aplikačního rámce – jeho analýza, návrh a implementace. Tento aplikační rámec je nakonec otestován v rámci implementace vzorové hry Méďa Béďa.

Klíčová slova: desková hra, aplikační rámec, C#, .NET Framework, Méďa Béďa

Abstract

The aim of this thesis is to analyze common features and principles of board games and design environment for design of these games. The thesis describes a board game as such, provides a brief history and their distribution. The thesis also describes the development environments/frameworks that are already developed and their most important properties. These properties serve as the basis for a new framework for board games. Further, thesis describes the development of this framework – the analysis, design and implementation. This framework is then tested in the implementation of a model Yogi Bear game.

Keywords: board game, framework, C#, .NET Framework, Yogi Bear

Seznam použitých zkratek a symbolů

DSL	–	Domain-Specific Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Již od nepaměti se člověk snaží nějak si zkrátit dlouhou chvíli, zpříjemnit si dlouhé zimní večery, nějak se zabavit ve chvílích volna, odpočinout si. Někteří lidé preferují aktivní odpočinek formou pohybu, tedy nějakého sportu, kutilství, domácích prací apod. Jiní lidé zase tráví svůj volný čas pouhým sezením v restauračních zařízeních, u televize či u počítače. Konečně, další skupina lidí preferuje krácení volného času hraním společenských her, konkrétně her deskových.

Deskové hry odjakživa patřily mezi vyhledávaný prvek společenské zábavy. Lidé se při jejich hraní nejen dobře baví a využívají volný čas, ale především procvičují mnoho činností mozku (logickou, kognitivní, paměťovou atd.) a někdy také pohybové dovednosti.

V dnešní době existuje nepřeberné množství různých deskových her a jejich variací. Od jednoduchých her na papíře po komplexní, velmi obsáhlé a náročné hry. Bohužel, ne každý si takovéto hry může dovolit. Některé hry hrané v jiných zemích nemusí být dostupné v zemích jiných. Některé hry, které jsme hráli v dětských letech, nemusí být dokonce vůbec k mání.

Řešením těchto a dalších problémů může být vytvoření počítačové hry jako alternativy ke hře deskové. Nemusíme tak řešit problémy opotřebení různých částí hry, jejich ztrátou, přenositelností, podváděním a mnoho dalších. Můžeme si upravit různé části hry k obrazu svému nebo dokonce vytvořit svou novou hru podle vlastních představ a fantazie.

Nápadů na tvorbu takovýchto počítačových her může být mnoho. Pokaždé však mají tvorby těchto her něco společného. Může se jednat o stejné figurky, stejné karty, kostky a mnoho dalších naprosto stejných prvků. Tvůrci takových her tak pokaždé musí znova a znova programovat ty samé věci místo toho, aby se soustředili na podstatnější části tvorby.

Cílem této práce tedy bude zanalyzovat společné rysy a principy deskových her a navrhnout prostředí (aplikační rámec) pro návrh těchto her a usnadnit tak jejich tvorbu, oprostit ji od vytváření částí, které již jednou vytvořeny byly.

Práce bude rozdělena na pět částí. První část bude pojednávat obecně o deskových hrách, jejich rozdělení, společných prvcích a principech. Ve druhé části se podíváme na již vytvořené aplikační rámce a jejich nejdůležitější vlastnosti. Třetí část práce bude soustředěna na analýzu, návrh a implementaci našeho aplikačního rámce. Ve čtvrté části otestujeme námi vytvořený rámec v rámci implementace vzorové deskové hry. V poslední části zhodnotíme výsledky naší práce, tedy čeho jsme dosáhli a čeho by bylo vhodné dosáhnout v budoucnu.

2 Deskové hry

Deskovou hru bychom mohli charakterizovat jako hru, jejíž hlavním znakem je deska, hrací plocha s plánem hry, na kterém se tato hra hraje. Používají se zde kameny či figury, které jsou na desku umisťovány, jsou z ní odstraňovány nebo jsou po desce posouvány podle předem daných pravidel.

Styl hraní těchto her bývá založen na strategii (šachy) nebo na náhodě (hod kostkou), popř. na jejich kombinaci. Deskové hry mají také většinou stanoven nějaký cíl, kterého se hráči snaží dosáhnout.

2.1 Historie deskových her

Deskové hry mají dlouhou a velmi zajímavou historii, provázejí člověka více než 5 000 let. Zde si ale pouze nastíníme historii deskových her ve formě přibližné časové osy [1]:

- 3 500 př. n. l. – Senet (nález v hrobkách v Egyptě)
- 3 000 př. n. l. – Mehen (nález v Egyptě, hrací kameny měly tvar lvů)
- 2 560 př. n. l. – Královská hra z Uru (na území bývalého Sumeru nalezeny hrací desky)
- 2 500 př. n. l. – malby her Senet a Han
- 2 000 př. n. l. – malba v hrobce v Benihassanu – 2 neznámé hry (o jedné se předpokládá, že byla určena pro hru Tau)
- 1 500 př. n. l. – hra Liubo vytesána na trámu z modrého kamene
- 1 400 př. n. l. – na střeše chrámu Kurna vyřezány desky na hry Alquerque, Mlýnek, Mlýn a pravděpodobně Mancala
- 200 př. n. l. – čínská deska na Go, která pochází z tohoto období, je uložena v muzeu v Pekingu (nicméně hra go je mnohem starší)
- 116–27 př. n. l. – dílo Marcuse Terentius Varra nazvané Lingua Latina X (II, par. 20) obsahuje nejstarší zmínku o hře latrunculi
- 79–8 př. n. l. – dílo Shuo yuan Liu Xianga obsahuje nejstarší zmínku o hře Xiangqi
- 8 n. l. – v Ovidiově díle Umění milovati je nejstarší zmínka o hře Ludus Duodecim Scriptorum (předchůdce Vrhcábů)
- 5. století – v severní Evropě se hrají hry rodiny Tafl (Vlci a ovce, Tablut. . .)
- 7. století – nejstarší známá zmínka o hře Čaturanga (předchůdce hry Šachy) v sanskrtské básnické sbírce Vasavadatta, kterou napsal Subandhu
- 1 888 – vynalezena hra Reversi (původní název hry dnes známé také jako Othello)

2.2 Dělení deskových her

Druhů deskových her je nepřeberné množství, není tedy jednoduché je jednoznačně kategorizovat. Z různých variant dělení deskových her podle různých autorů si vybereme pouze ta nejčastěji popisované dělení.

2.2.1 Historické dělení

Deskové hry jsou rozděleny na dvě kategorie:

- **Klasické deskové hry** – do této kategorie bychom mohli zařadit převážně staré deskové hry, u kterých neznáme autora a jsme schopni určit nanejvýš období, ve kterém vznikly. Jedná se např. o hry Dáma, Šachy, Go, Mahjong. . .
- **Moderní deskové hry** – zde patří především nové hry, u kterých ve většině případů známe jak autora, tak přesný rok jejich vzniku. Jedná se o hry jako Monopoly, Dostihy a sázky, Logik apod.

2.2.2 Dělení podle stylu hry

Zapletal [2] také rozděluje deskové hry na čtyři základní druhy:

- **Strategické hry** – mají za úkol zajmout, zablokovat či obklíčit protihráče, zabrat větší část hrací desky, tedy souboj dvou či několika soupeřů (Šachy, Dáma).
- **Závodivé hry** – zde mají hráči za cíl dostat se jako první na určené místo, nasbírat větší počet bodů či nasbírat rychleji určitý počet bodů (Člověče, nezlob se!, Dostihy a sázky).
- **Poziční hry** – smyslem hry je manévrování na desce za účelem docílení určité sestavy (Mlýn, Piškvorky).
- **Pátrací hry** – smyslem hry je hledání, řešení nějakého problému (Potápěči).

2.2.3 Další dělení deskových her

Předchozí dvě dělení rozhodně nebyla vyčerpávající, uveďme si ještě ve zkratce další obvyklá dělení deskových her:

- Abstraktní
- Vzdělávací
- Historické
- Válečné
- Slovní
- Atd.

3 Frameworky pro deskové hry

Abychom mohli popisovat vývoj frameworku (aplikačního rámce) pro deskové hry, musíme nejdříve vědět, co slovo framework znamená. Jedná se o softwarovou strukturu, která slouží jako podpora při programování softwarových projektů. Jeho cílem je převzetí typických problémů dané oblasti a usnadnit návrhářům vývoj projektu tak, aby se mohl soustředit pouze na své zadání.

Frameworků pro vývoj různých typů aplikací je celá řada a nejinak je tomu u frameworků pro deskové hry. Inspirací pro tuto práci byly dva frameworky: FlexibleRules a Vassal.

3.1 FlexibleRules

Disertační práce studenta Frapolli [3]. Jedná se o komplexní framework, který umožňuje uživatelům vytvářet počítačové deskové hry. Jádro frameworku tvoří konceptuální model, který usnadňuje návrh deskových her napodobením přístupu užívaného k vysvětlení struktury hry živému hráči. Na základě tohoto modelu byl zaveden DSL k implementaci těchto her. Je zde obsaženo také vývojové prostředí pro lepší produktivitu koncových uživatelů, především těch se základní znalostí programování.

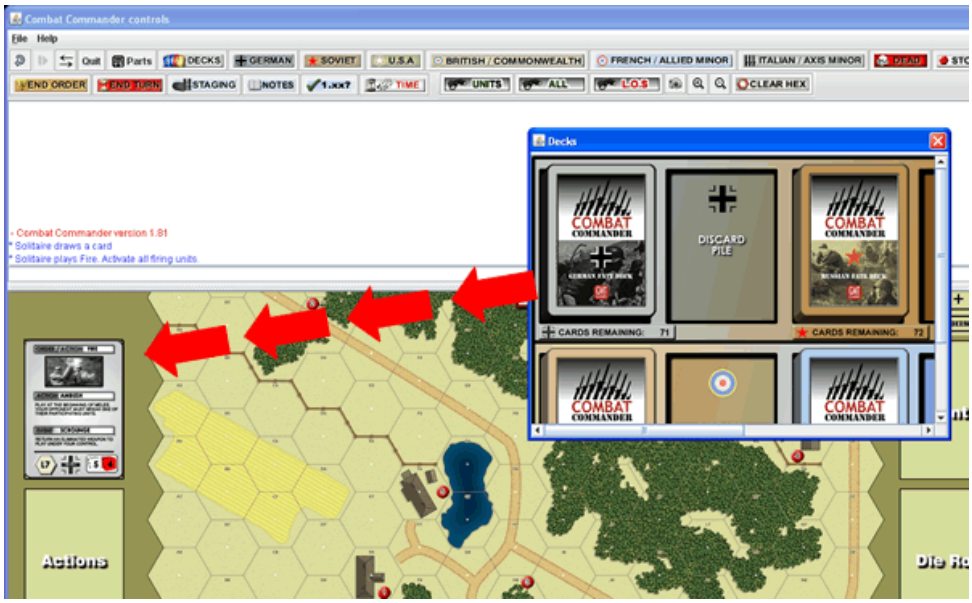
Obrázek 1: Ukázka prostředí frameworku FlexibleRules (Zdroj: [3])

3.2 Vassal

Vassal [4] je open-source nástroj pro tvorbu počítačových deskových her. Pomocí něj můžeme vytvořit mnoho typů her (válečné, karetní apod.). Hráči mohou hrát se spoluhráči (protihráči) online na Vassal serveru, přes peer-to-peer spojení, emailem nebo off-line.

Vassal podporuje tvorbu těchto druhů her (a mnoha dalších):

- Tradiční deskové hry.
- Karetní hry.
- RPG využívající taktickou mapu.



Obrázek 2: Ukázka prostředí frameworku Vassal (Zdroj: [4])

3.3 Global GameSpace

V případě Global GameSpace [5] se nejedná přímo o framework pro deskové hry, ale je zde uvedeno velké množství informací, které jsou potřeba k sestrojení jakékoliv deskové hry. Tento web však pohlíží na tvorbu her z materiálního hlediska, tzn. vývoj deskové hry jako přesné kopie ke hře skutečné (krabice, ve které je určitý počet figurek, určitý obnos peněz v mincích či bankovkách).

3.4 Další frameworky pro deskové hry

Frapolli ve své práci [3] uvádí seznam dostupných frameworků pro deskové hry v různých programovacích jazycích. Některé z frameworků však nejsou zaměřeny pouze na deskové hry:

Název	Jazyk	Web
Allegro	C/C++	http://alleg.sourceforge.net/
LWJGL	Java	http://lwjgl.org/
Pygame	Python	http://www.pygame.org/
Rubygame	Ruby	http://rubygame.org/
AgateLib	.NET	http://www.agatelib.org/
PLib	C++	http://plib.sourceforge.net/
Sphere	Javascript	http://www.spheredev.org/
vbGore	Visual Basic	http://www.vbgore.com/
Amaltheia	C/C++	http://home.gna.org/amaltheia/
OpenTK	C#/Mono	http://www.opentk.com/
Pyglet	Python	http://www.pyglet.org/
Squeak	Smalltalk	http://www.squeak.org/
XNA Game Studio	.NET	http://www.xna.com/

Tabulka 1: Frameworky pro deskové hry

3.5 Shrnutí vlastností frameworků

Nyní si shrňme nedůležitější vlastnosti výše uvedených frameworků, které by námi vyvíjený framework měl mít také. Jedná se o:

- Možnost vytvořit prakticky jakoukoliv deskovou hru.
- Poskytnout co největší počet prostředků pro vývoj deskových her.
- Možnost přiřazení jakémukoliv objektu grafické vlastnosti (velikost, pozici, pozadí).
- Dát uživateli možnost kontroly neobvyklých situací (např. sejmutí karty z prázdného balíčku). V případě nedodržení daných pravidel vyvolání výjimky.
- Předpřipravit uživateli některá hotová řešení (např. vytvoření cesty z políček).

Uveďme si také některé velmi užitečné vlastnosti, kterými se však zabývat nebudeme (alespoň prozatím):

- Podpora online hry.
- Grafické prostředí pro návrh her.
- DSL pro snadnější implementaci.

4 Analýza, návrh a implementace rámce

4.1 Specifikace požadavků

Dříve, než budeme cokoliv programovat, musíme specifikovat, jaké požadavky by námi vytvářený framework měl splňovat. Splňovat nejen z hlediska funkčnosti, ale také z hlediska technického.

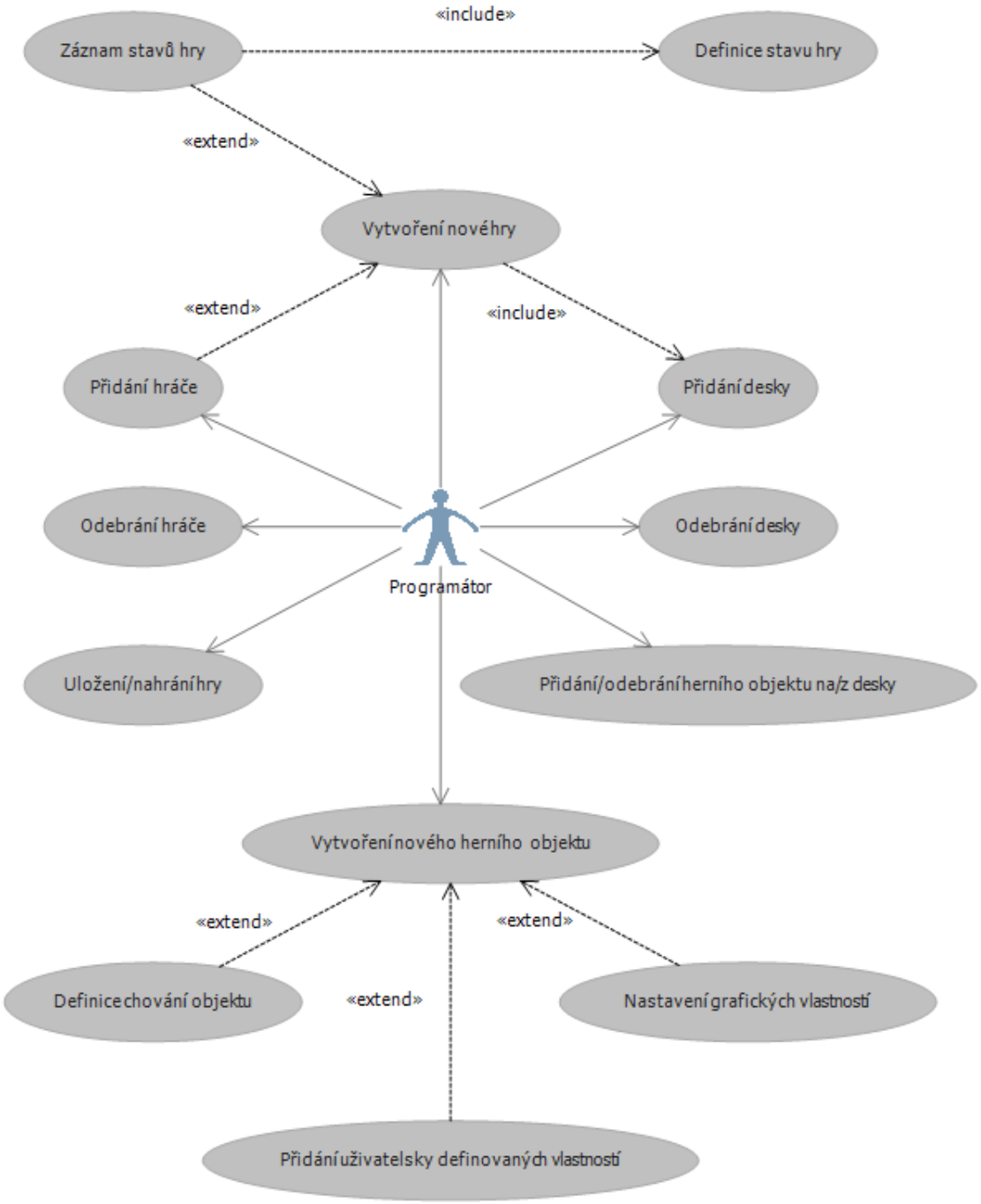
V předchozí kapitole jsme si uvedli nejdůležitější vlastnosti již vytvořených frameworků. Kromě těchto vlastností by náš framework měl navíc splňovat ještě následující:

- Možnost ukládání jednotlivých stavů hry dle definice uživatele.
- V případě vytváření herního plánu se neomezovat pouze na určité typy objektů (např. pouze čtvercová pole se čtyřmi sousedy, nemožnost vytvoření vlastního herního plánu), jak je tomu u některých frameworků.
- Neomezovat se pouze na určitý typ her (např. karetní hry).
- Možnost zakomponování dalších prvků hry, které nejsou ve frameworku implementovány (speciální předměty, vlastnosti hráče apod.).

Dále je třeba si uvědomit, že ne všechny prvky hry je vhodné převést do elektronické podoby jako jejich přesnou kopii (jak je tomu na Global GameSpace [5], kde je popisována např. implementace krabice od hry, ve které se nacházejí nevyužité herní prvky). Některé prvky dokonce není vhodné (nebo je zbytečné) převádět vůbec.

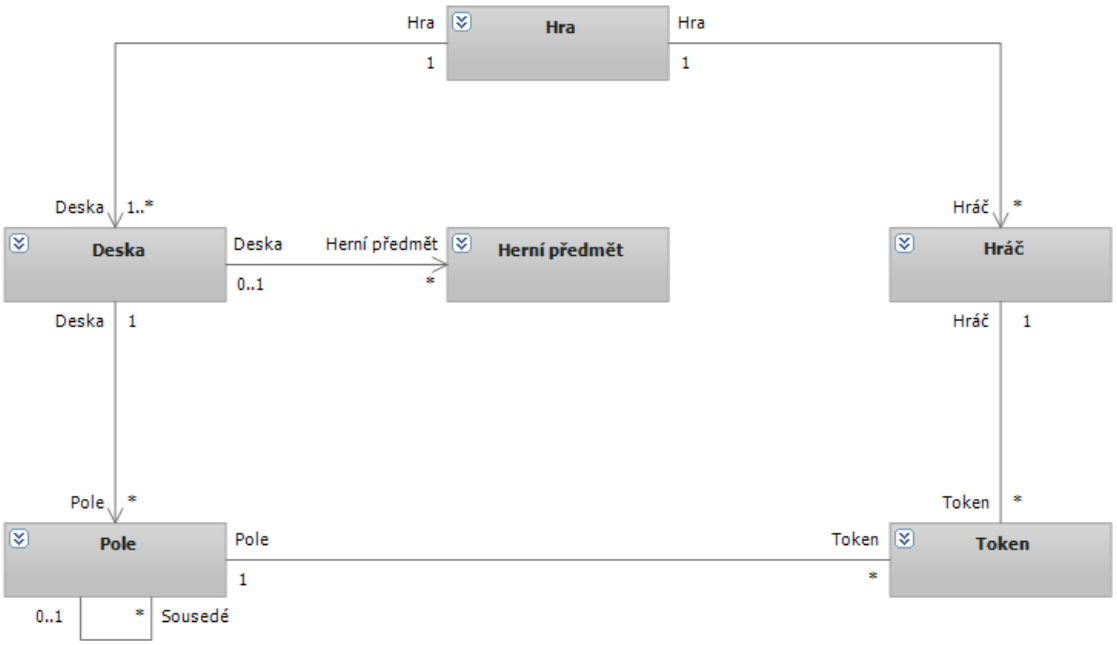
Framework bude implementován v jazyce C# pod frameworkem .NET verze 4.0 jako Class Library (knihovna tříd). Hry vytvářené pomocí tohoto frameworku tak nebudou muset být nutně implementované v jazyce C#. Je také jedno, jaké uživatelské rozhraní bude pro hry vytvořeno (může se jednat např. o konzolovou aplikaci, webovou aplikaci či Windows Forms aplikaci).

Uveďme si zde také diagram případů užití systému. Jediným aktérem zde bude programátor, koncový uživatel, který bude framework využívat pro tvorbu deskových her.



Obrázek 3: Diagram případů užití systému

4.2 Konceptuální model frameworku



Obrázek 4: Konceptuální model frameworku

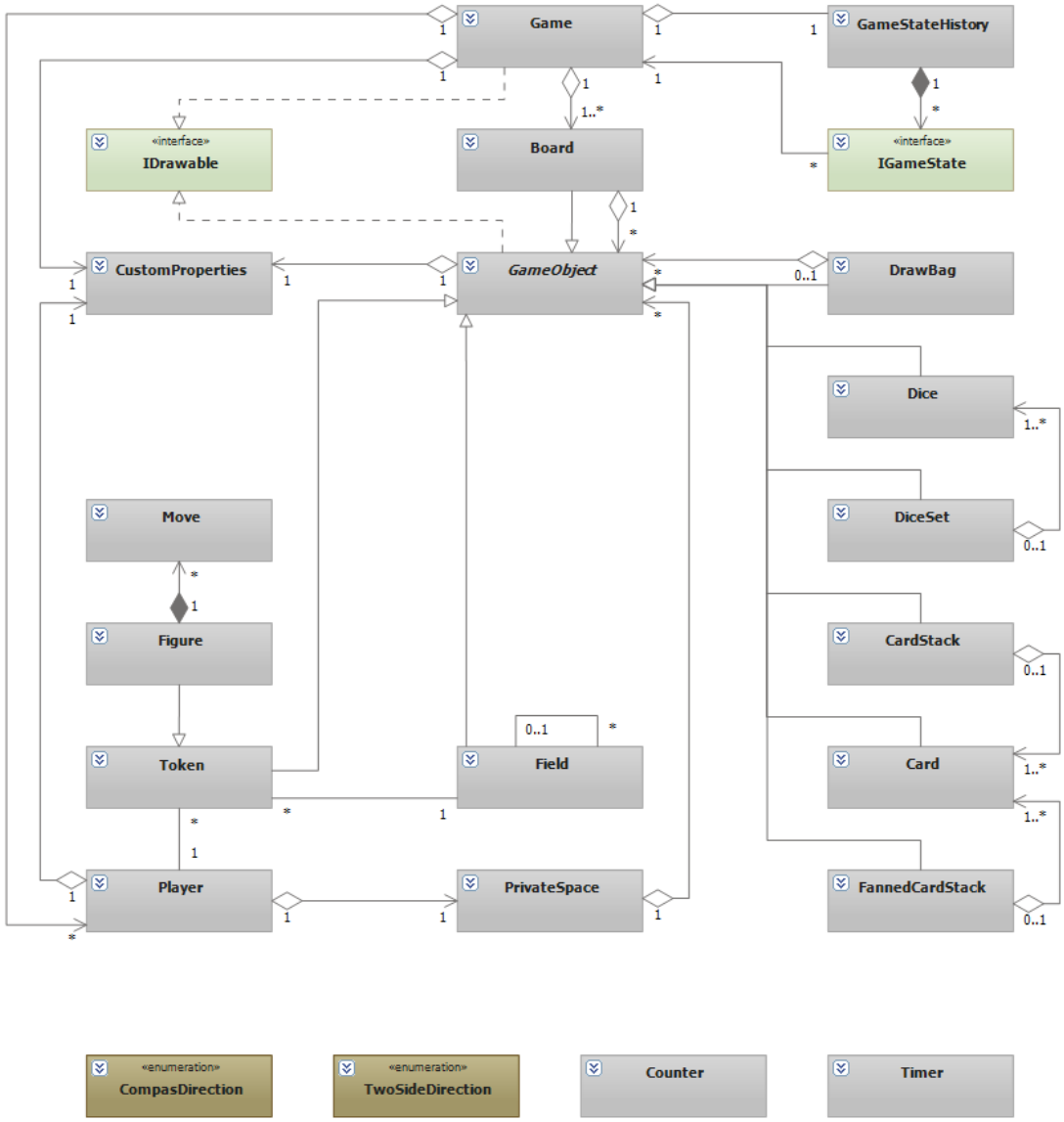
Uvedený konceptuální model zobrazuje základní (a velmi obecnou) strukturu frameworku, ze které budeme při jeho návrhu vycházet.

Základní entitou je hra, kterou si můžeme fyzicky představit jako stůl, kolem kterého sedí určitý počet hráčů. Na tomto stole se nachází minimálně jedna hrací deska, na které je nakreslen herní plán.

Tento plán se skládá z polí. Každé pole může mít definované sousedy, na které je možno přeskočit. Na těchto polích jsou umísťovány tzv. Tokeny. Jedná se o libovolné, žetony, značky nebo především o herní figurky, jež jsou ve vlastnictví určitého hráče.

Na herní desce se můžou nacházet kostky či karty, tedy různé herní předměty.

4.3 Doménový model frameworku



Obrázek 5: Doménový model frameworku

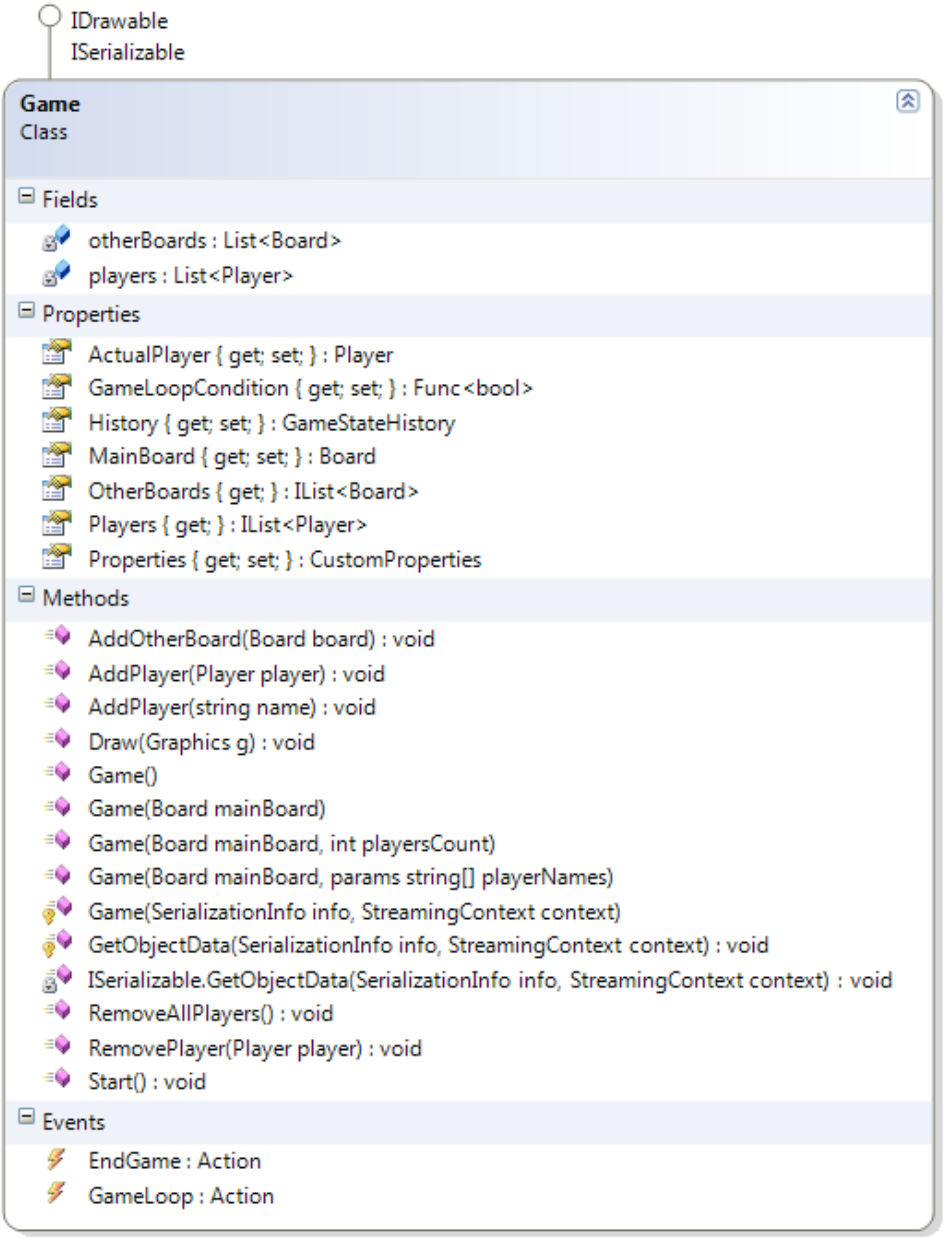
4.4 Popis jednotlivých tříd frameworku

Abychom si o frameworku udělali nějakou představu, musíme vědět, z jakých tříd se bude skládat a k čemu dané třídy slouží. Zde jsou nastíněné základní funkce daných tříd frameworku. Kompletní výčet tříd, jejich funkcí a dokumentace je uveden v příloze.

4.4.1 Rozhraní IDrawable

Třída, která bude implementovat toto rozhraní, bude umět vykreslit svůj obsah metodou `Draw` pomocí třídy `Graphics`.

4.4.2 Třída Game



Obrázek 6: Třída Game

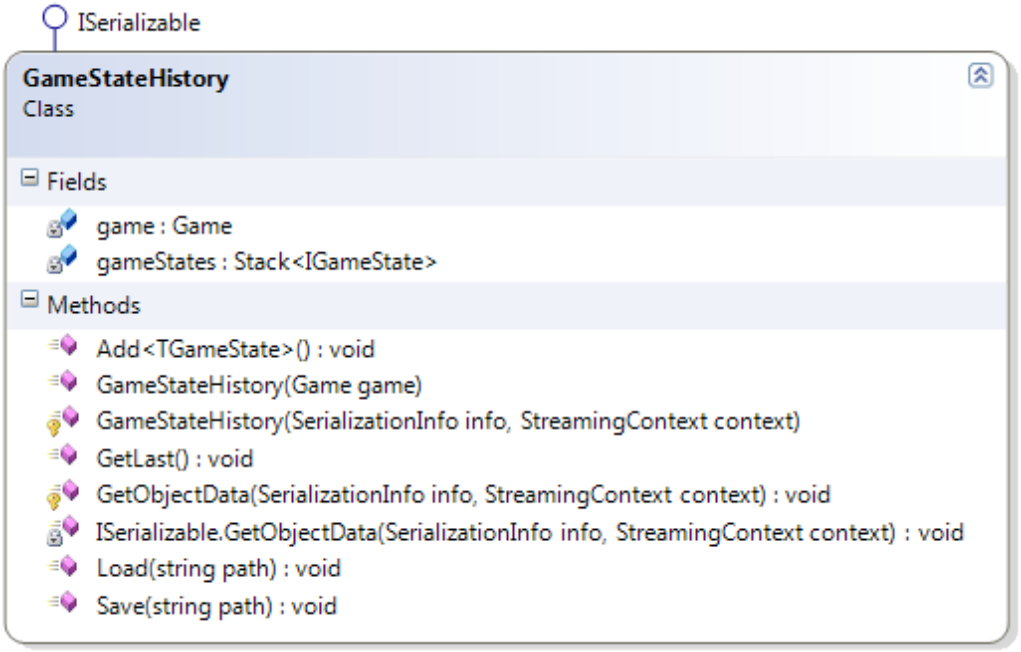
Hra bude evidovat hráče, herní desky. Jeden hráč bude označen jako aktuální a jedna herní deska bude hlavní. Bude zde také evidence stavů hry – herní historie. Můžeme zde také přidat uživatelsky definované vlastnosti.

Instanci nové hry je implicitně vytvořena s jednou prázdnou hlavní deskou. Lze také vytvořit instanci hry s předpřipravenou deskou. Můžeme rovněž zadat jména hráčů nebo pouze jejich počet.

Do hry budeme moci přidat hráče či herní desku, které můžeme ze hry i smazat. Můžeme smazat také všechny hráče najednou.

Před spuštěním hry je potřeba nadefinovat průběh hry. Tento průběh bude s největší pravděpodobností cyklický, proto musíme ještě nadefinovat podmínku opakování tohoto cyklu. Na závěr je třeba nadefinovat akci, která se provede po ukončení průběhu hry.

4.4.3 Třída GameSateHistory



Obrázek 7: Třída GameStateHistory

Jedná se o vlastnost hry, která uchovává jednotlivé stavy hry v zásobníku. Tyto stavy se neukládají automaticky, jejich ukládání řídí uživatel. Stav hry lze jak uložit, tak následně nahrát zpátky do hry.

Námi vytvořený framework definuje pouze rozhraní, které musí implementovat třída, která bude sloužit pro ukládání stavů hry a kterou musí nadefinovat sám uživatel. Musí mít také bezparametrický konstruktor, jelikož instance této třídy se vytváří přímo při ukládání stavu hry:

```
public void Add<TGameState>()
    where TGameState : IGameState, new()
{
    TGameState state = new TGameState();
    state.Create(game);
    gameStates.Push(state);
}
```

Výpis 1: Funkce pro přidání stavu hry do zásobníku

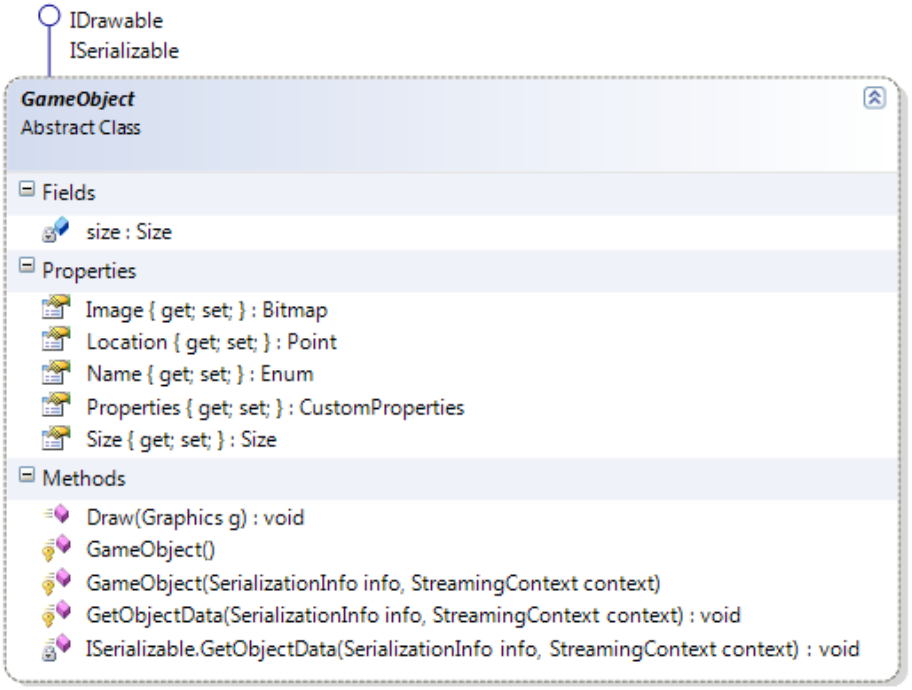
Pro ukládání všech stavů hry do souboru je použita binární serializace objektu. Tento soubor je uložen na zadanou cestu. Po opětovném načtení tohoto souboru je načten poslední uložený stav do hry.

4.4.4 Rozhraní IGameState

Odvozené třídy budou umět vytvořit a uchovat uživatelsky definované informace o hře, které budou ukládány. Tyto informace pak bude možné nahrát zpátky do hry.

Třída, která bude implementovat toto rozhraní, by také měla mít bezparametrický konstruktor a měla by být označena atributem `System.SerializableAttribute`.

4.4.5 Třída GameObject

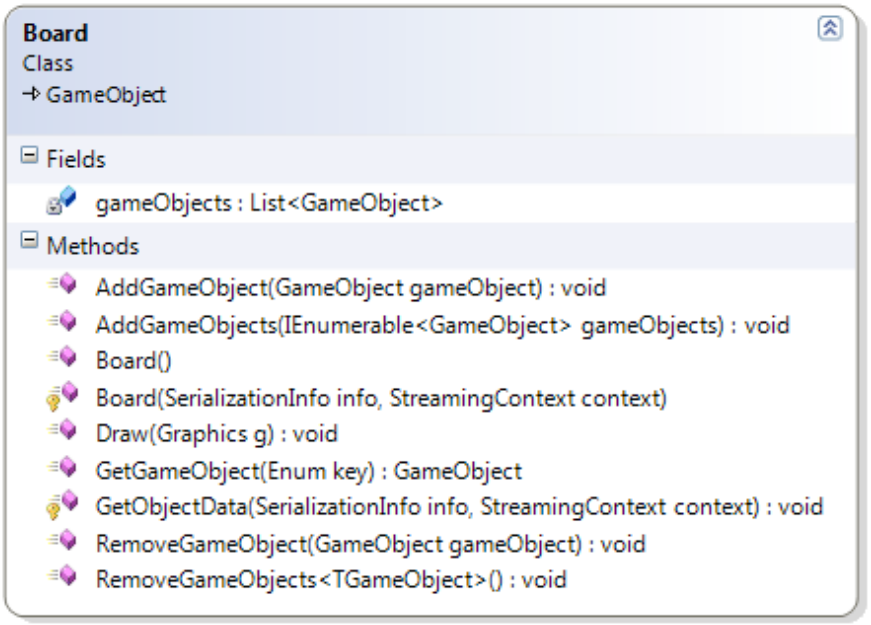


Obrázek 8: Třída GameObject

Třída, která definuje všechny společné vlastnosti (název, pozici, velikost, obrázek na pozadí) a také vlastnosti uživatelsky definované pro jakýkoliv herní objekt. Společné vlastnosti jsou potřeba zejména při vykreslování objektu. Pokud nezadáme velikost objektu, bude použita velikost obrázku, je-li definován.

Tato třída je abstraktní, je navržena pouze jako základ nových herních objektů, nelze tak vytvořit její instanci.

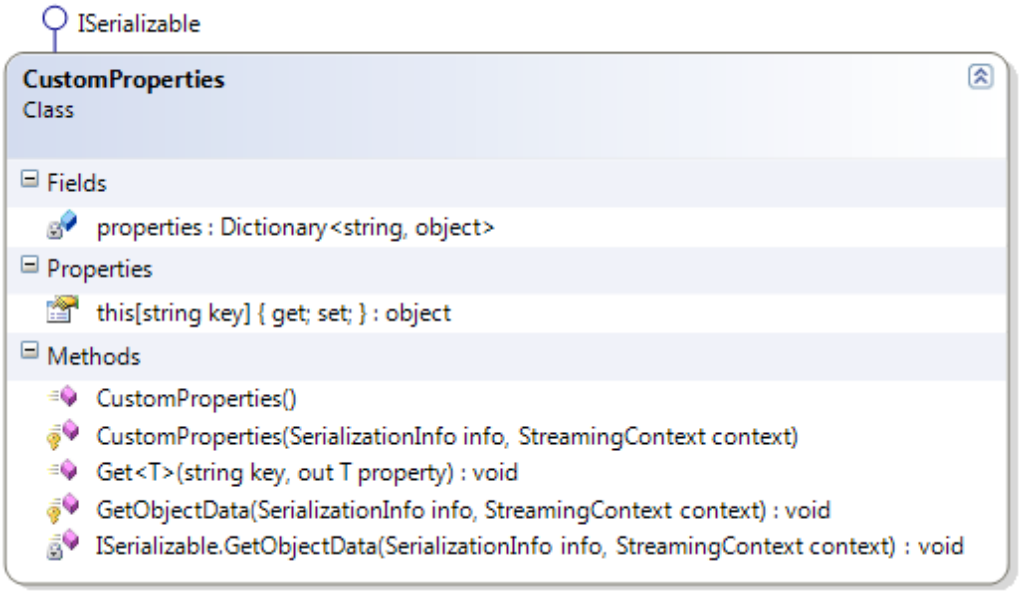
4.4.6 Třída Board



Obrázek 9: Třída Board

Reprezentace herní desky. Jediné, co tato deska bude uchovávat, je kolekce herních objektů. Tyto objekty můžeme na desku přidat nebo je z desky odebrat. Můžeme dokonce odebrat všechny objekty určitého typu (např. všechny figurky). Pokud máme některé objekty pojmenované, můžeme je tak snadno z desky získat.

4.4.7 Třída CustomProperties



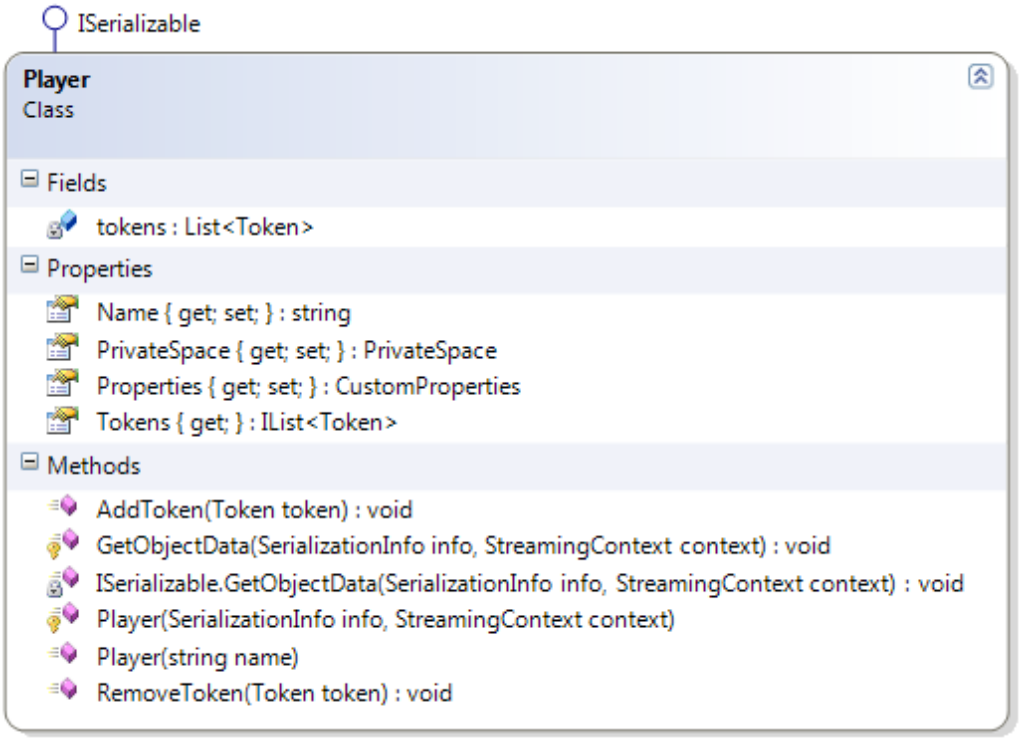
Obrázek 10: Třída CustomProperties

Uživatelsky definované vlastnosti. Jedná se o kolekci objektů představujících vlastnosti. S instancí této třídy pracujeme obdobně jako s polem – vlastnosti jsou identifikovány svým názvem. Pokud přiřazujeme hodnotu vlastnosti, která ještě nebyla vytvořena, vytvoří se automaticky.

```
figure.Properties["previousField"] = null;
Field field = figure.Properties["previousField"];
```

Výpis 2: Ukázka práce s instancí třídy CustomProperties

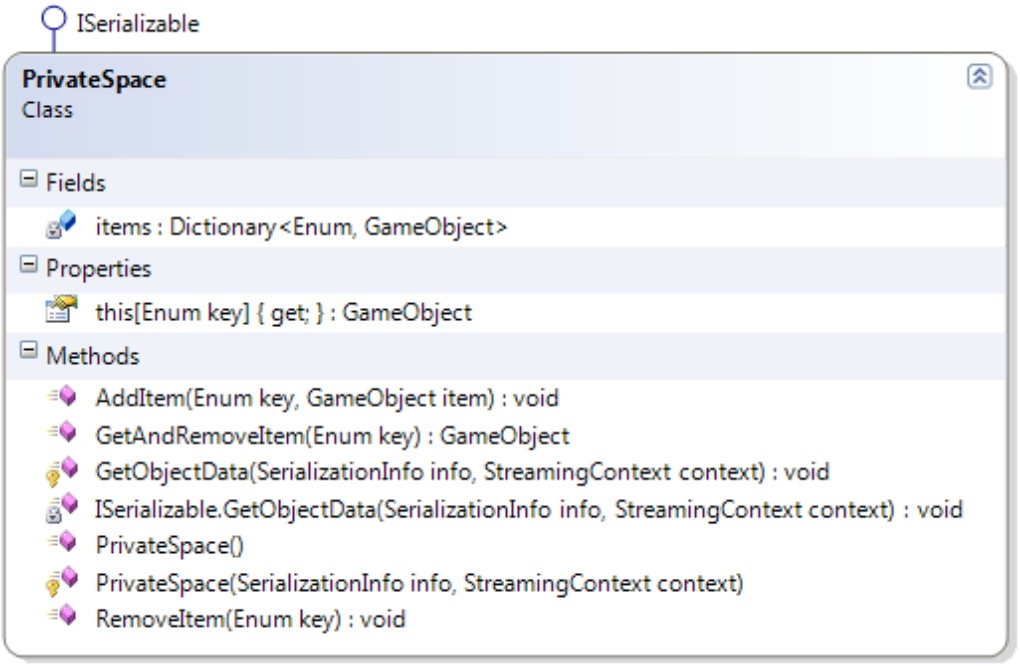
4.4.8 Třída Player



Obrázek 11: Třída Player

Reprezentace pojmenovaného hráče. Tento hráč má ve vlastnictví určité množství Tokenů, které můžeme libovolně přidávat či odebírat. Uživatel také může hráči definovat libovolné vlastnosti. Každý hráč má také k dispozici soukromý prostor.

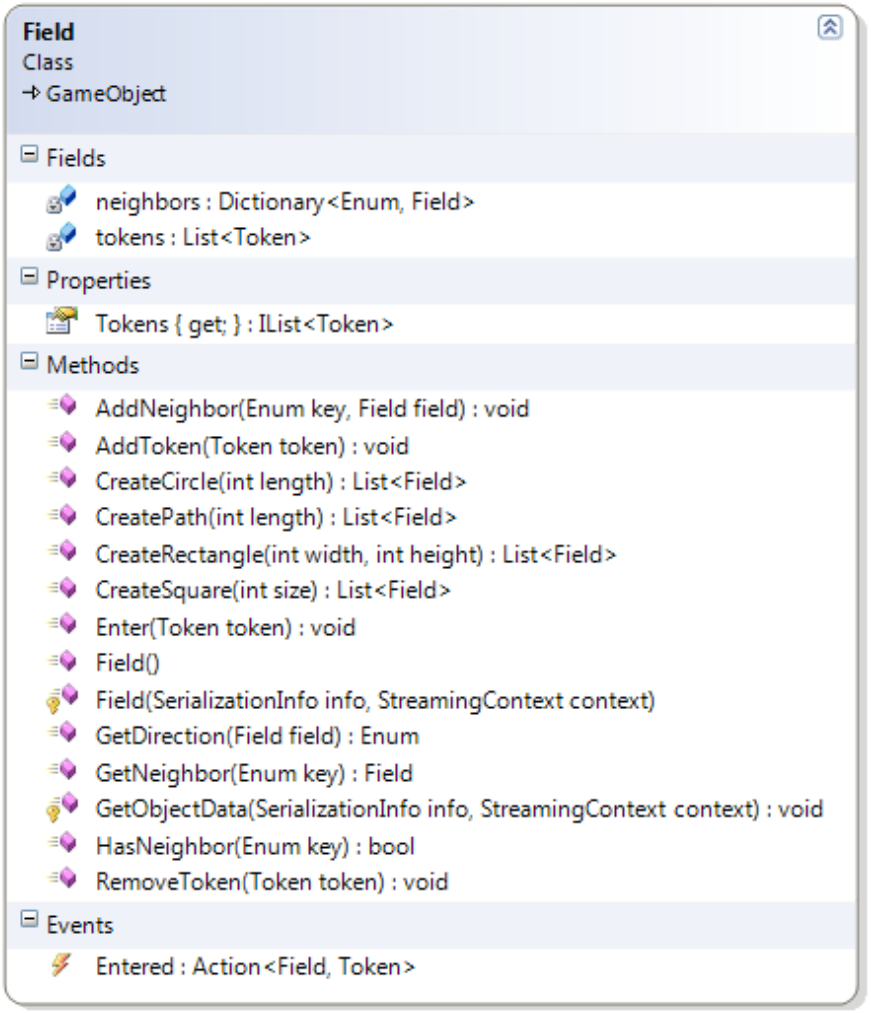
4.4.9 Třída PrivateSpace



Obrázek 12: Třída PrivateSpace

Jedná se o soukromý prostor, kde může odkládat věci (herní objekty) a skrýt je tak před ostatními hráči (příkladem užití mohou být karetní hry). Tyto objekty jsou jednoznačně identifikovány. Můžeme je libovolně přidávat a odebírat.

4.4.10 Třída Field



Obrázek 13: Třída Field

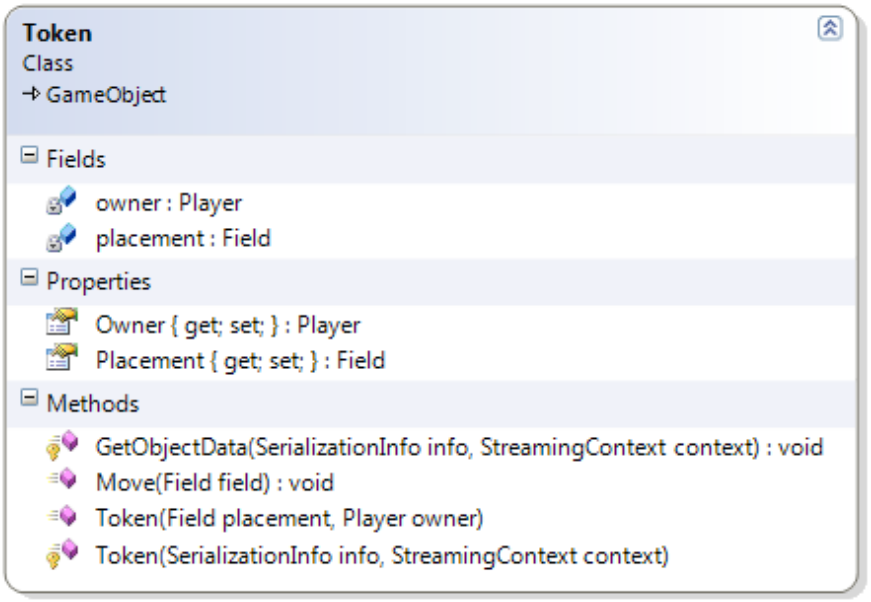
Tato třída reprezentuje herní políčko. Pro vytvoření herního plánu na herní desce je důležité tato políčka vytvořit tak, aby bylo možné přecházet z jednoho na druhé podle předem daných pravidel. Toto je realizováno seznamem sousedních políček pod jednoznačnými identifikátory, které každé políčko obsahuje. Můžeme pak získat odkaz na políčko souseda nebo naopak získat směr, kterým je třeba se k sousedovi vydat.

Na políčku může být umístěno určité množství Tokenů. Tyto tokeny můžeme libovolně přidávat a odebírat. Tokeny, především figurkami, v průběhu hry nějak pohybujeme – přemísťujeme je z políčka na políčko. Můžeme tedy nadefinovat akci, která se provede po vstupu Tokenu na políčko.

Abychom koncovému uživateli alespoň trochu ušetřili práci, jsou zde definovány metody, které vrací předpřipravené seskupení správně propojených polí. Jedná se o:

- **Cestu** – jednoduchá cesta z políček, která má oba konce bez sousedů.
- **Kruh** – obdoba cesty, zde jsou ale oba konce spojeny dohromady.
- **Obdélník** – obdélník z polí, která jsou vzájemně propojena.
- **Čtverec** – obdoba obdélníku se stejnými rozměry (např. šachovnice).

4.4.11 Třída Token

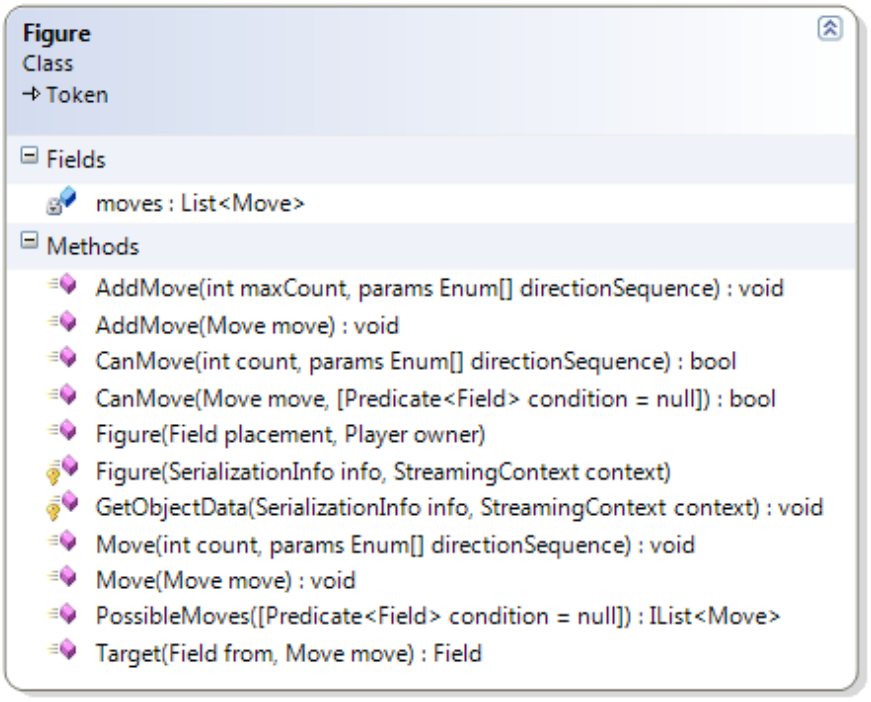


Obrázek 14: Třída Token

Token si můžeme představit jako libovolný žeton, figurku nebo nějakou značku. Tento žeton se vyznačuje tím, že je vždy umístěn na nějaké herní políčko a vždy je také ve vlastnictví určitého hráče. Při změně hráče je starému hráči žeton odebrán a novému naopak přidán. Obdobně je tomu i při přesunu žetonu z jednoho pole na druhé.

Jediné, co s tímto žetonem můžeme v průběhu hry dělat, je jeho přesunutí z jednoho herního políčka na druhé. Při přesunu žetonu se zároveň nastavuje jeho nová poloha. Ta je implicitně nastavena na střed políčka, na které je žeton přesunut.

4.4.12 Třída Figure



Obrázek 15: Třída Figure

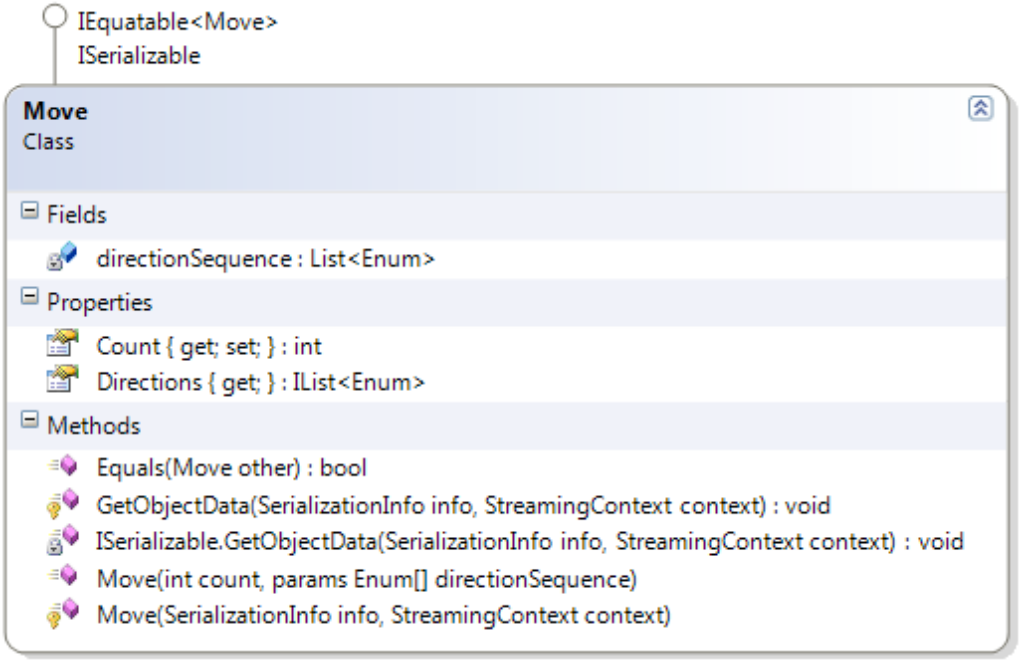
Speciálním případem třídy `Token` je třída `Figure` neboli herní figurka. Tato figurka rozšiřuje Token o možnost definice přesných pohybů, které může figurka provádět.

Pohybů může být libovolné množství. Abychom předešli zbytečné práci uživatele při definici stejných pohybů, které se liší jen počtem jejich opakování (např. pohyby figurky věže ve hře šachy), stačí uživateli zadat jen základní pohyb (např. doprava) a maximální počet jeho opakování.

Pro konkrétní figurku na konkrétním místě lze také zjistit všechny možné pohyby nebo zjistit, zda daný pohyb je proveditelný. Můžeme také zjistit případný cíl, na který se figurka dostane po provedení určitého pohybu.

Při zjišťování možných pohybů také můžeme definovat podmínku, podle které dále se rozhodne, zda je daný pohyb proveditelný nebo ne.

4.4.13 Třída Move



Obrázek 16: Třída Move

Třída `Move` představuje pohyb herní figurky. Tento pohyb je složen ze zadané sekvence směrů, kterým se má figurka posouvat, a počtu jejich opakování. Je také možné porovnat dva pohyby a zjistit, zda si odpovídají.

Použití typu `System.Enum` jako identifikátoru směrů pohybu je mnohem lepší než např. použití typu `System.String`. Je zde kupř. vyloučena možnost, že se uživatel přepíše.

Použití tohoto typu ale přináší jedno úskalí – serializace. Ukládání stavu hry sice proběhne bez problému, ale při následné deserializaci objektu je vyhozena výjimka `System.MemberAccessException`. Tento problém způsobuje právě položka datového typu `System.Collections.Generic.List<System.Enum>`. Jedním z možných řešení tohoto problému je implementace metody `GetObjectData` deklarované rozhraním `System.Runtime.Serialization.ISerializable`, pomocí které uložíme seznam směrů pohybu položku po položce. Při deserializaci objektu podobným způsobem seznam směrů pohybu rekonstruujeme.

```
protected Move(SerializationInfo info, StreamingContext context)
{
    Count = info.GetInt32("count");
    int count = info.GetInt32("countList");
    directionSequence = new List<Enum>(count);
    for (int i = 0; i < count; i++)
    {
```

```
            directionSequence.Add((Enum)info.GetValue("direction" + i, typeof(Enum)));
        }
    }

    protected virtual void GetObjectData(SerializationInfo info, StreamingContext context)
    {
        info.AddValue("count", Count);
        info.AddValue("countList", directionSequence.Count);
        for (int i = 0; i < directionSequence.Count; i++)
        {
            info.AddValue("direction" + i, directionSequence[i]);
        }
    }

    void ISerializable.GetObjectData(SerializationInfo info, StreamingContext context)
    {
        GetObjectData(info, context);
    }
```

Výpis 3: Ukázka implementace rozhraní ISerializable

4.4.14 Třída DrawBag

Jedná se o pytlík, do kterého můžeme vložit herní objekty určeného typu. Vyjmout z pytlíku můžeme ale pouze jen náhodně vybraný objekt. Vždy máme přehled o počtu vložených objektů.

4.4.15 Třída Dice

Jedná se o hrací kostku. Tato kostka může mít libovolný počet stran a na každé z těchto stran bude hodnota námi zadaného typu, nemusíme se tedy omezovat pouze na čísla. Při přidávání jedné strany a její hodnoty můžeme zadat pravděpodobnost, se kterou může být daná hodnota vylosována. Ke každé hodnotě může být přiřazen jeden obrázek.

Kostkou hodíme metodou `Roll`. Přitom se náhodně vybere jedna z množiny stran a spustí se akce, která má být při hodu kostkou spuštěna, pokud je definována.

4.4.16 Třída DiceSet

Soubor kostek, který můžeme znát např. ze hry Liar's dice. Je to množina kostek stejného typu. Při jeho vytváření stačí zadat kostku, která se bude používat jako vzor pro tvorbu kostek dalších. Při přidávání kostek tak stačí uvést pouze jejich počet.

Zavoláním metody `Roll` hodíme všemi vloženými kostkami a díky implementaci rozhraní `System.Collections.Generic.IEnumerable` pak můžeme procházet postupně všechny kostky a zjišťovat jejich vržené hodnoty.

4.4.17 Třída Card

Reprezentace karty, která nese pouze hodnotu námi zadaného typu. Zobrazením hodnoty karty můžeme rovněž vyvolat akci, která má být pro zobrazení hodnoty vyvolaná. Vkládané hodnoty musí implementovat rozhraní `System.IComparable`.

4.4.18 Třída CardStack

Balíček karet stejného typu. Do tohoto balíčku můžeme přidávat karty jak shora, tak zespoda. Stejným způsobem můžeme také karty z balíčku odebírat. Nemusíme se přitom omezovat pouze na jednu kartu, přidat či odebrat můžeme balíček s určitým počtem karet.

Balíček můžeme zamíchat, tedy náhodně rozmístit karty v něm uložené. Pro nejrůznější účely lze balíček také seřadit. Seřadit lze buď defaultním porovnáváním hodnot, nebo vlastní implementací porovnávání.

4.4.19 Třída FannedCardStack

Obdoba třídy `CardStack`, taktéž se svým způsobem jedná o balíček karet. Oproti třídě `CardStack` zde máme přístup ke všem kartám najednou („vějíř“ z karet). Kartu můžeme položit na libovolné místo a z libovolného místa můžeme také kartu sejmout. Všechny karty můžeme také složit do podoby klasického balíčku karet.

4.4.20 Třída Counter

Jednoduchý čítač, který umí buď o 1 svou hodnotu přičítat, nebo o 1 odečítat. Můžeme mu také nastavit počáteční hodnotu a koncovou hodnotu počítání. Po dosažení koncové hodnoty je možné provést nějakou uživatelem definovanou akci.

4.4.21 Třída Timer

Jednoduchý časovač. Jeho tiky jsou v sekundových intervalech. Základní funkcí je počítání času od nuly. Další funkcí je odečítání času od uživatelem zadané hodnoty a při dosažení nuly provedení uživatelem definované akce. Toto lze provádět i opakovaně (např. každých 10 sekund hodit kostkou).

4.4.22 Enumerátory TwoSideDirection a CompasDirection

Jedná se o předpřipravené výčtové typy, které jsou použity k propojování herních políček.

Enumerátor `TwoSideDirection` je použit ve statických metodách `CreatePath` a `CreateCircle` třídy `Field`.

Enumerátor `CompasDirection` je použit ve statických metodách `CreateSquare` a `CreateRectangle` třídy `Field`.

5 Testování rámce

Nyní, když máme framework pro deskové hry vytvořený, nám zbývá tento framework otestovat. Jeho testování provedeme tím, že pomocí něj implementujeme vzorovou deskovou hru. Tímto rámec nejen otestujeme a zjistíme tak jeho případné nedostatky (které napravíme), ale také tak budoucím uživatelům rámce ukážeme, jak s tímto rámcem pracovat, jak pomocí něj nějakou hru implementovat.

5.1 Ukázková hra

Hra, kterou budeme v rámci testování tohoto frameworku implementovat, se nazývá Méďa Béďa. Jedná se o společenskou hru pro 2 – 4 hráče na motivy stejnojmenného seriálu. Dle rozdělení deskových her v kapitole 1 bychom tuto hru mohli charakterizovat jako moderní závodivou deskovou hru.

5.1.1 Cíl hry

Cílem hry je figurkami co nejrychleji projít po cestičkách v parku od brány na poslední pole se smajlíkem. Cestou musí každý sebrat návštěvníkům parku jeden piknikový košík. Na některých polích plní figurky různé úkoly.

5.1.2 Pravidla hry

Každý hráč si vybere jednu figurku a napíše vedle ní své jméno. V pořadí, v jakém byla jména zapsána, hází hráči kostkou s čísly a postupují se svou figurkou po cestičkách v parku. Prochází-li hráč křižovatkou, musí si hodit barevnou kostkou. Barva mu přikáže směr, kterým bude dál pokračovat v postupu. Avšak pozor! Padne-li hráči černá, zůstává stát na křižovatce a v příštím kole bude házet barevnou kostkou i kostkou s čísly najednou. Postoupí pak podle barvy a počtu bodů.

Každý hráč musí získat během cesty k cíli jeden piknikový košík. Ten získá, pokud padne přímým hodem na pole s košíkem. Pokud již jeden košík má, další si nebere. Konečné pole a pole očíslovaná musíte dosáhnout jen přímým hodem. Hráč, který dosáhne konečné pole bez košíku, pokračuje ve hře. Všechna tato zvláštní pole jsou oku skrytá.

Hra probíhá tak dlouho, dokud jsou ve hře alespoň dva hráči.

5.1.3 Příslušenství hry

Každá desková hra je souborem jednotlivých předmětů, které mají při jejím hraní určitý význam. Může se jednat o plán hry, figurky, peníze apod. V případě hry Méďa Béďa to jsou:

- 1 krabice
- 1 herní plán
- 4 papírové figurky

- 4 podstavce k figurkám
- 4 karty
- 1 kostka s čísly
- 1 barevná kostka
- 1 návod ke hře

5.2 Poznámky k implementaci hry

5.2.1 Skutečná hra vs. její elektronická reprezentace

V kapitole 3 je uvedeno, že „ne všechny prvky hry je vhodné převést do elektronické podoby jako jejich přesnou kopii.“ Nejinak tomu je v případě hry Méďa Béďa. Následující tabulka popisuje, v jaké formě budou příslušné prvky hry reprezentovány:

Herní prvek	**Reprezentace prvku**
Krabice	Neimplementováno
Herní plán	Třídy `Board` a `Field`
Figurky	Třída `Figure`
Podstavce k figurkám	Neimplementováno
Karty (vlastnictví košíku)	Uživatelsky definovaná vlastnost hráče
Kostky	Třída `Dice`
Návod ke hře	Zvláštní formulář

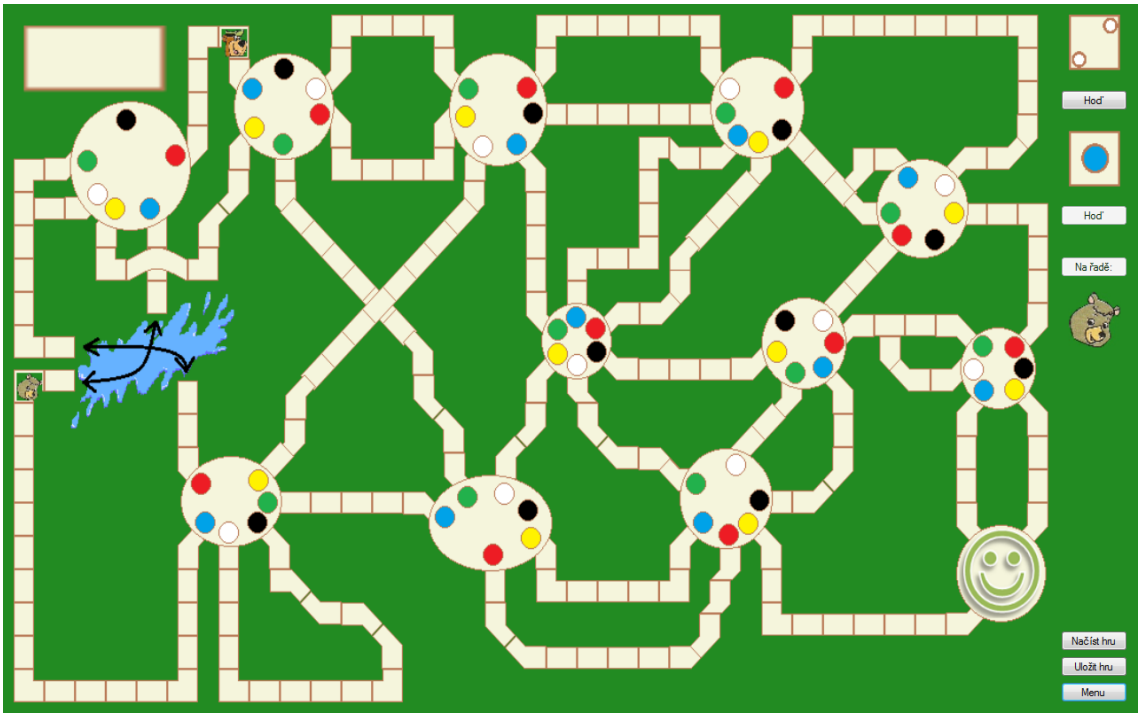
Tabulka 2: Reprezentace skutečných herních prvků ve hře

5.2.2 Uživatelské rozhraní

Uživatelské rozhraní hry bude realizováno pomocí Windows Forms. Důraz bude také kladen na přehlednost a hlavně jednoduchost rozhraní, tj. minimální množství použitých komponent. Čím více „zbytečného“ kódu by v aplikaci bylo, tím těžší by bylo pro budoucí uživatele frameworku zorientovat se v něm a zaměřit se pouze na jeho stěžejní prvky.

Rozhraní hry bude rozdělené na čtyři části:

1. **Hlavní menu** – jednoduché menu skládající se ze čtyř tlačítek pro zahájení nové hry, načtení uložené hry, zobrazení nápovědy a ukončení hry.
2. **Nová hra** – nastavené počtu hráčů a přiřazení obrázků postavičky ke jménu hráče.
3. **Hra** – nejdůležitější obrazovka. Zde je zobrazen herní plán, na kterém jsou umístěny postavičky. Jsou zde také zobrazeny kostky, se kterými budou hráči házet, a také ovládací prvky hry.
4. **Konec hry** – zde je zobrazeno pořadí, v jakém hráči skončili hru.



Obrázek 17: Ukázka hry

5.3 Implementace hry

5.3.1 Využití extension metod

Námi vytvořený framework zcela jistě nepokrývá reprezentaci všech možných prvků deskových her. Nejinak je tomu i u hry Méďa Béďa. Jako částečné řešení tohoto problému se jeví implementace třídy `CustomProperties`.

Ukládání těchto vlastností a jejich zpětné načítání se však neobejde bez přístupu do kolekce (zadáním klíče ve formě řetězce) a následného přetypování objektu na požadovaný datový typ. Výsledný kód tak nejen nevypadá dobře, ale hlavně při mnohonásobném použití stejného příkazu se vystavujeme riziku nesprávného zápisu názvu vlastnosti, nehledě na možnou potřebu změny názvu vlastnosti či chybného přetypování.

Dalším případem neefektivního způsobu programování může být např. přiřazení více vlastností najednou k mnoha objektům (v našem případě grafické vlastnosti políčka na herním plánu).

Řešením těchto problémů je využití tzv. extension metod. Jedná se o metody, které se tváří jako instanční metody dané třídy. Nemusíme tak odvozovat nový datový typ od stávajícího pouze kvůli jedné metodě.

U hry Méďa Béďa těchto metod využijeme k nastavování a získávání vlastností hráče (zda vlastní košík, zda jedno kolo stojí), pro zjištění předchozího políčka figurky, ke změně aktuálního hráče hry apod.

```
public static Figure GetFigure(this Player player)
{
    return (Figure)player.Tokens[0];
}

public static void HasBasket(this Player player, bool value)
{
    player.Properties["hasBasket"] = value;
}

public static bool HasBasket(this Player player)
{
    return (bool)player.Properties["hasBasket"];
}

public static void SetImage(this GameObject field, Bitmap image, Point location, RotateFlipType
      rotation, Color transparentColor)
{
    field.Image = image;
    field.Location = location;
    field.Image.RotateFlip(rotation);
    field.Image.MakeTransparent(transparentColor);
}
```

Výpis 4: Ukázka využití extension metod

5.3.2 Vlastní enumerátory

Určitě také pro vývoj hry nebudou stačit pouze dva enumerátory obsažené ve frameworku. Bude potřeba nějak pojmenovat speciální herní políčka nebo určité herní objekty, se kterými budeme chtít později pracovat, nějak identifikovat sousedy určitého políčka apod.

V našem případě to budou např. hodnoty barevné kostky (tím pádem i identifikátorů sousedních políček na křižovatkách), startovní či koncové políčko, identifikace kostek a identifikace hráče (Béďa, Bubu. . .).

Z důvodu případné binární serializace při ukládání stavu hry by měly být všechny takto vytvořené enumerátory označeny atributem `System.SerializableAttribute`.

5.3.3 Reprezentace stavu hry

Pokud budeme ve hře chtít ukládat stavy hry, bude nutné vytvořit novou třídu, která implementuje rozhraní `IGameState`, má bezparametrický konstruktor a je označena atributem `System.SerializableAttribute`.

Stav naší hry bude reprezentován třídou GameState, která uchovává tyto informace:

- Aktuální hráč
- Výsledné pořadí hráčů

- Informace o všech hráčích (realizováno seznamem uspořádaných šestic):
  - Hráč
  - Pole, na kterém se nachází hráčova figurka
  - Pole, na kterém se hráčova figurka nacházela v předchozím kole
  - Informace, zda hráč vlastní košík
  - Informace, zda hráč musí v příštím kole hodit 4, 5 nebo 6
  - Informace, zda hráč příští kolo stojí

Tyto informace bude možné, dle deklarace rozhraní, buď ze hry načíst, nebo zpátky do hry nahrát. Uložení aktuálního stavu hry do historie provedeme příkazem:

```
game.History.Add<GameState>();
```

Výpis 5: Uložení aktuálního stavu hry do historie

5.3.4 Ovládání Windows Forms Controls

V budoucnu budou grafická prostředí mnoha her určitě realizována pomocí Windows Forms. Určitě se také při jejich realizaci nevyhneme využití komponent odvozených od třídy `System.Windows.Forms.Control` (`Button`, `Label`, `TextBox` apod.). Jejich použití je jednoduché, problém však nastává při jejich ovládání z jiného vlákna, než ve kterém byly vytvořeny. Pak je vyvolána výjimka `System.InvalidOperationException`.

Na tuto skutečnost je třeba myslet především při ovládání těchto komponent z metody `GameLoopCondition`, události `GameLoop` a události `EndGame` třídy `Game`. Jsou totiž spouštěny v novém vlákně.

Ve hře Méďa Béďa budeme tímto stylem ovládat dvě tlačítka. První pro hod číselnou kostkou a druhé pro hod kostkou barevnou. Následující kód ukazuje možné řešení problému ovládání komponent z jiného vlákna:

```
public static void Enable(this Button button, bool value)
{
    if (button.InvokeRequired)
    {
        button.Invoke(new Action(() =>
        {
            button.Enabled = value;
        }));
    }
    else
    {
        button.Enabled = value;
    }
}
```

Výpis 6: Ovládání tlačítka z jiného vlákna

5.3.5 Implementace samotné hry

Nyní už máme vše připraveno k tomu, abychom mohli začít vytvářet samotnou hru. Předpokládejme také, že máme vytvořeny veškeré formuláře (menu, nová hra, konec hry a základní kostru hry). Vytvořme instanci nové hry, která implicitně obsahuje prázdnou hlavní herní desku. Nejdříve bude potřeba hru inicializovat.

Začněme vytvořením herního plánu. Ten se v našem případě skládá z cestiček a křižovatek. Křižovatka bude klasické políčko. Na tvorbu cestiček můžeme použít metodu `CreatePath` třídy `Field`. Všechny tyto vytvořené objekty přidáme na hlavní herní desku.

Na hlavní herní desce už sice máme vytvořené cestičky a křižovatky, ale budeme je muset propojit. To zařídíme metodou `AddNeighbor` třídy `Field`. Přitom si můžeme jednotlivá políčka pojmenovat. My pojmenujeme startovní a koncové pole. Dále bychom měli každému políčku a také samotné desce přiřadit grafické vlastnosti: obrázek, pozici a rotaci. Na to už máme připravenou metodu `SetImage`.

Další důležitou věcí je přiřazení akcí, které se provedou po vstupu na dané políčko. Pokud chceme na více různých políček nastavit stejnou akci, můžeme vytvořit funkci, která tuto akci vykoná. Pokud je daná akce specifická pouze pro jedno políčko, stačí použít lambda výrazu.

```
void FieldWithBasketEnter(Field field, Token token)
{
    MessageBox.Show("Vstoupil jsi na pole s piknikovým košíkem, jeden si dostal.", "Košík",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
    token.Owner.HasBasket(true);
}
r1g_r9g[5].Entered += FieldWithBasketEnter;

r7y_r11ur[1].Entered += (field, token) =>
{
    MessageBox.Show("Pan strážce tě přistihl s košíkem a ty ho jdi vrátit. Přejdi za pole 5.", "
        Vstup na pole č. 20", MessageBoxButtons.OK, MessageBoxIcon.Information);
    token.Owner.HasBasket(false);
    ((Figure)token).Move(r2w_r8y[9]);
    ((Figure)token).PreviousField(r2w_r8y[10]);
};
```

Výpis 7: Definování akcí po vstupu na políčko

V následujícím kroku vytvoříme dvě kostky. Jednu číselnou a druhou barevnou (hodnoty budou stejného typu, jaký identifikuje sousední políčka na křižovatkách). Těmto kostkám definujeme obrázky na pozadí pro každou hodnotu (stranu) zvlášť. Přiřadíme jim také chování – pouze nastavení proměnné a vykreslení hry. Tlačítkům pro hody kostkami přiřadíme akci, která kostkou hodí (zde vidíme smysl pojmenování určitých herních objektů):

```
((Dice<DiceColor>)(game.MainBoard.GetGameObject(SpecialGameObject.ColorDice))).Roll();
```

Výpis 8: Hod kostkou

Na vykreslení hry definujeme jednoduchou metodu, která vykreslí jednak postavičku aktuálního hráče, ale především zavolá metodu `Draw` na instanci třídy `Game`. Ta pak vykreslí všechny objekty přidané do hry.

Dále je nutné přidat do hry hráče. Z formuláře Nová hra obdržíme seznam uspořádaných dvojic: jméno hráče a herní postavička. Do hry tak postupně budeme přidávat jednotlivé hráče ze seznamu podle jména. Při vytváření jim nastavíme všechny požadované vlastnosti. Poté hráči přiřadíme novou postavičku. Tato postavička bude mít definované všechny své možné pohyby a také své pozadí (obrázek je vybrán dle herní postavičky vybrané v menu). Nakonec přidáme hráče do hry a postavičku na herní desku.

Nakonec je potřeba definovat podmínku opakování herního cyklu, samotný herní cyklus a akci, která se provede po jeho skončení. Tím máme hru vytvořenou, můžeme ji spustit a hrát. Případné uložení čí načtení hry je velmi jednoduché (v případě načítání hry je nutné ji následně překreslit):

```
game.History.Save(”somePath”);

game.History.Load(”somePath”);
DrawGame();
```

Výpis 9: Uložení a načtení hry

6 Závěr

V této bakalářské práci jsme se zabývali vývojem elektronických (počítačových) deskových her. Seznámili jsme se s jejich základními společnými prvky a principy. Ukázali jsme si již vytvořené nástroje pro vývoj těchto her.

Cílem této práce bylo zanalyzovat společné rysy a principy deskových her a navrhnout prostředí pro návrh těchto her. Výstupem práce je v tomto případě knihovna tříd, ve které jsou definovány třídy potřebné pro návrh deskových her. Díky této knihovně tříd se bude moci budoucí vývojář deskových her soustředit pouze na vývoj hry jako takové, ne na neustále opakovaný vývoj společných částí.

Tato práce pokrývá zatím pouze jen implementaci společných prvků a jejich chování. Podporuje také základní definici grafických vlastností objektů. Díky tomu můžeme sestrojit jakoukoliv deskovou hru, jejíž obsah můžeme vykreslovat. Implementace ukázkové hry tak byla velice jednoduchá, stačilo jen minimální „doprogramování “.

Přestože byla implementace ukázkové hry jednoduchá, velice zdlouhavé bylo nastavování grafických vlastností objektů, především políček (obrázek, otočení, průhledná barva, pozice), kterých je na desce přes 200.

V budoucnu by tedy rozhodně bylo dobré rozšířit tento framework o grafické rozhraní – vývojové prostředí. Vývojáři deskových her by tak ušetřili čas a vývoj hry by byl mnohem pohodlnější. Pro snazší vývoj her a zpřístupnění vývoje her méně zkušeným/začínajícím programátorům by také bylo vhodné zavést DSL. Dalším prvkem frameworku, který by bylo vhodné v budoucnu zavést, je podpora online hraní. Hru tak nebude muset hrát mnoho hráčů u jednoho počítače. Dále má tato funkce velký význam např. u karetních her, u kterých je zapotřebí soukromí.

Cíl práce je tedy, alespoň podle mě, splněn, přestože na vývoji frameworku je ještě mnoho práce. Díky této práci jsem si také prohloubil své znalosti jazyka C#, .NET frameworku a programovacích technik. Nabyté zkušenosti zcela jistě využiju při tvorbě budoucích projektů.

7 Reference

[1] LADA. *Klub přátel deskových her* [online]. 19. 10. 2009 [cit. 2012-04-20]. Dostupné z: http://www.deskovehry.info/historie.php

[2] ZAPLETAL, Miloš. *Velká kniha deskových her*. Praha: Mladá fronta, 1991. ISBN 80-204-0188-4.

[3] FRAPOLLI, Fulvio. *FlexibleRules: A Player Oriented Board Game Development Framework* [online]. Fribourg, 12.10.2010 [cit. 2012-04-1]. Dostupné z: http://data.fulviofrapolli.net/Thesis/FulvioFrapolliThesis.pdf. Disertační práce. University of Fribourg (Švýcarsko).

[4] KINNEY, Rodney. *Vassal: The open-source boardgame engine* [online]. 2009 [cit. 2012-04-1]. Dostupné z: http://www.vassalengine.org

[5] LACY, Curtis. *Global GameSpace* [online]. 2012 [cit. 2012-04-1]. Dostupné z: http://www.globalgamespace.org

A Obsah DVD

Následující tabulka popisuje umístění souborů na DVD a jejich popis.

Adresář	**Popis**
/doc/framework	Dokumentace frameworku
/doc/thesis	Text bakalářské práce
/src/BoardGameFramework	Visual Studio 2010 Project - Framework
/src/YogiBear	Visual Studio 2010 Project - Ukázková hra

Tabulka 3: Obsah DVD